

# **Kryptographie - eine mathematische Einführung**

Rosa Freund <rosa@pool.math.tu-berlin.de>

28. Dezember 2004

# Überblick

- Grundlegende Fragestellungen
- Symmetrische Verschlüsselung: Blockchiffren, Hashfunktionen
- asymmetrische Verschlüsselung: RSA, Diskretes Logarithmus Problem (DLP), DLP auf elliptischen Kurven

## Grundlegende Fragestellungen

- Eine Nachricht soll nur vom vorgesehenen Empfänger gelesen werden können. Es geht nicht darum, das Lauschen zu verhindern, sondern darum, daß Lauscher die Nachricht nicht entschlüsseln können
- Die Signatur einer Nachricht soll es jedem ermöglichen, die Identität des Absenders zu verifizieren. Gleichzeitig soll es unmöglich sein, eine gefälschte Nachricht mit gültiger Signatur zu erstellen

## Allgemeines

Zum verschlüsselten Kommunizieren benötigen die Parteien

- ein Verschlüsselungsverfahren, mit dem ver- und entschlüsselt wird (z.B. PGP, SSL)
- einen Schlüssel, d.h. den Variablen, die das Verfahren benötigt, müssen Werte zugeordnet werden
- z.B. Verschlüsselungsverfahren Cäsarchiffre, Schlüssel ist die Zahl, um die das Alphabet verschoben wird

## Symmetrische Verfahren

- Die kommunizierenden Parteien teilen sich einen geheimen Schlüssel (secret key)
- z.B. Blockchiffren, Stromchiffren, Hashfunktionen
- Problem: Schlüsseltausch

## Asymmetrische Verfahren 1

- Diffie und Hellman, 1976: *New Directions in Cryptography*
- Es gibt einen öffentlichen sowie einen geheimen Schlüssel (public key, private key), der geheime ist schwer oder gar nicht aus dem öffentlichen berechenbar
- Um verschlüsselte Nachrichten erhalten bzw. Nachrichten signieren zu können, generiert A sich einen öffentlichen und einen geheimen Schlüssel
- Wie der Name sagt, muß A ihren geheimen Schlüssel (private key) geheimhalten, den öffentlichen (public key) jedoch veröffentlichen

## Asymmetrische Verfahren 2

- Beim Verschlüsseln nutzt B den öffentlichen Schlüssel von A, um eine Nachricht an A zu verschlüsseln. A entschlüsselt die Nachricht mit ihrem geheimen Schlüssel
- Beim Signieren signiert A die Nachricht mit ihrem geheimen Schlüssel. B benutzt As öffentlichen Schlüssel, um die Signatur zu verifizieren

## Asymmetrische Verfahren 3

- Sicherheit beruht meist auf algorithmischen Problemen mit hoher (bzw. ungeklärter) Komplexität
- Mathematisch geht insbesondere algebraische Zahlentheorie ein (z.B. diskreter Logarithmus, elliptische Kurven)
- Häufig zum Austausch von Schlüsseln für symmetrische Verfahren genutzt (z.B. SSL), da weniger effizient als symmetrische Verfahren

## Kerckhoff-Prinzip

- Auguste Kerckhoff, 1883: *La Cryptographie militaire*
- Sicherheit eines kryptographischen Systems sollte nur auf der Geheimhaltung des Schlüssels beruhen, nicht jedoch auf Geheimhaltung des Verfahrens selbst
- Vorteile: die Qualität des Verfahrens kann intensiver untersucht werden

## Blockchiffren

- symmetrisch
- Jede Nachricht wird in gleichlange Blöcke aufgeteilt, die Blöcke werden separat und unterschiedlich verschlüsselt. Der letzte Block wird ggf. mit Bits gepadded
- z.B. DES (1977), AES / Rijndael (2001)

## Hashfunktionen

- Berechnet für Inputs beliebiger Länge Hashwerte von vorgegebener (meist kurzer) Länge
- Geringe Änderung des Inputs führt zu stark geändertem Hashwert
- Unix-Paßworte werden als Hashes gespeichert
- Aber: Dictionary-Attacke

## Rechnen in Restklassen

- $\mathbb{Z}/q\mathbb{Z} := \{x + q\mathbb{Z} \mid x \in \mathbb{Z}\}$ , bzw.  $\mathbb{F}_q := \{0, 1, \dots, q - 1\}$
- Gerechnet wird modulo  $q$ , etwa wie bei einer Uhr (modulo 12)
- $q$  ist hier prim
- $\mathbb{F}_q$  ist ein Körper (d.h. Struktur mit  $+, -, *, /$ )

# RSA 1

- Rivest, Shamir und Adleman, 1977
- $p, q$  Primzahlen,  $n = pq$
- $n, d$  ist öffentlicher Schlüssel
- $e$  ist geheimer Schlüssel
- $\text{encrypt}(x) \equiv x^d \pmod{n}$
- $\text{decrypt}(y) \equiv y^e \pmod{n}$

## RSA 2

Zu zeigen:  $\text{decrypt}(\text{encrypt}(x)) = x$

- Wähle  $d$  mit  $1 < d < (p - 1)(q - 1) =: m$  und  $\text{ggT}(d, m) = 1$
- Wähle  $e$  mit  $1 < e < (p - 1)(q - 1)$  und  $ed \equiv 1 \pmod{m}$
- $\text{decrypt}(\text{encrypt}(x)) \equiv x^{de} \pmod{n}$ , also zeige  $x^{de} = x$  für  $x \in \mathbb{Z}/n\mathbb{Z}$
- Es ist  $ed = 1 + km = 1 + l(p - 1)$  nach Konstruktion
- Ferner gilt  $x^{p-1} = 1$  für  $x \in \mathbb{Z}/p\mathbb{Z}$  (Fermats kleiner Satz)

## RSA 3

- Also  $x^{ed} = xx^{l(p-1)} = x(x^{(p-1)})^l = x$  für  $x \in \mathbb{Z}/p\mathbb{Z}$
- Analog  $x^{ed} = x$  für  $x \in \mathbb{Z}/q\mathbb{Z}$
- Also auch  $x^{ed} = x$  für  $x \in \mathbb{Z}/pq\mathbb{Z} = \mathbb{Z}/n\mathbb{Z}$  (Chinesischer Restsatz:  $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \cong \mathbb{Z}/pq\mathbb{Z}$ )

## RSA 4

- Sicherheit von RSA beruht auf der Schwierigkeit des Faktorisierens von großen ganzen Zahlen  $n$
- In polynomieller Zeit ist lösbar: Ist  $n$  Primzahl?
- Vermutlich nicht in polynomieller Zeit lösbar: Was sind die Primfaktoren von  $n$ ?

## Diskretes Logarithmus Problem - DLP

- Wieder Rechnen in Restklassen
- Gegeben:  $g$  aus einer zyklischen Gruppe,  $x \in \mathbb{Z}$ . Gesucht:  $e$  mit  $g^e = x$
- z.B.  $\mathbb{Z}/11\mathbb{Z}$ : Suche  $e \in \mathbb{Z}/11\mathbb{Z}$  mit  $7^e \equiv 1 \pmod{11}$ . Lösung: 8.
- Verfahren z.B. ElGamal
- Mathematische Fragestellungen: In welchen Gruppen ist das DLP "besonders schwer"?

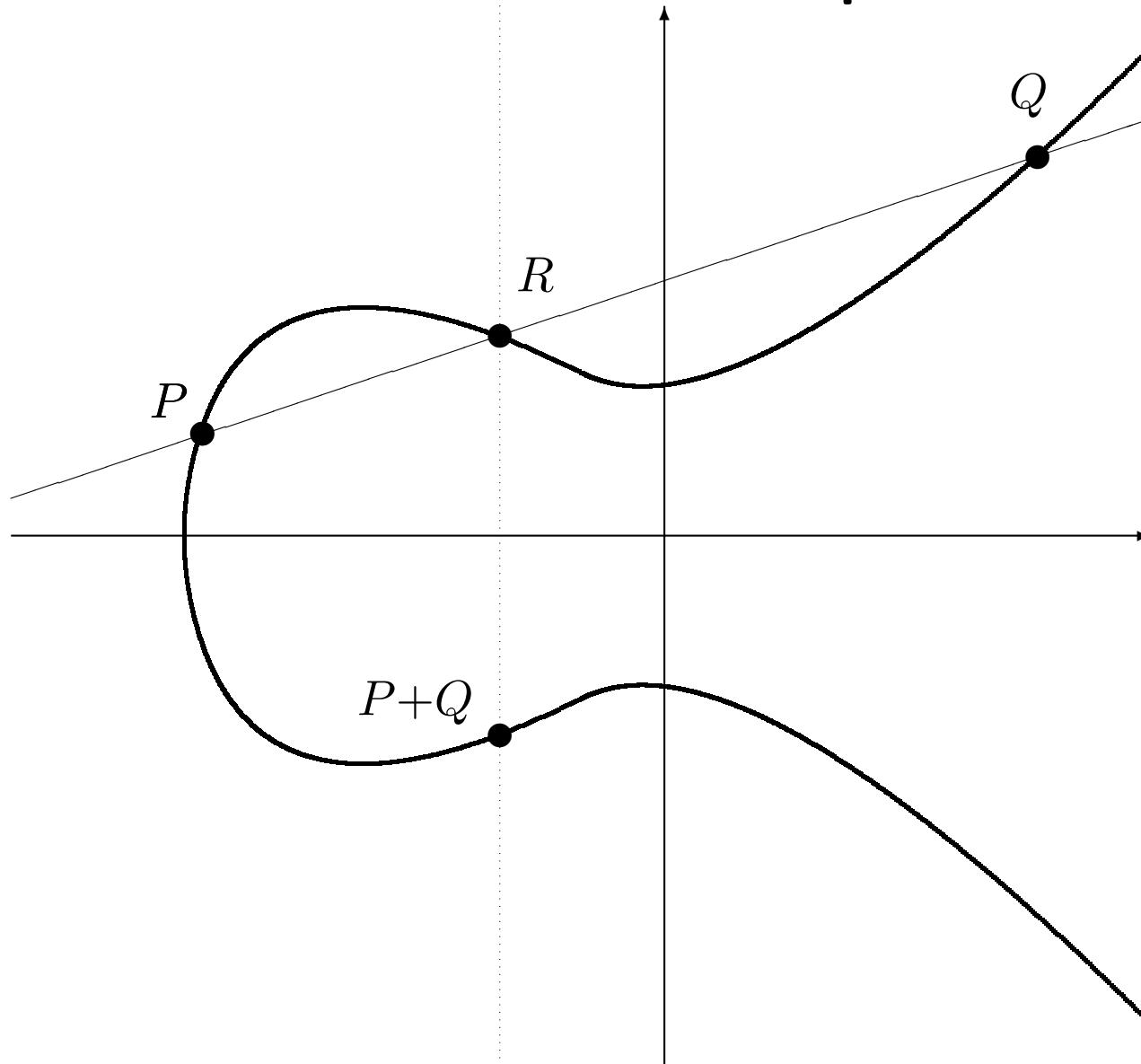
## EIGamal Verschlüsselung

- Sei  $\#G = l$ ,  $G$  zyklisch mit  $G = \langle g \rangle$ ,  $x \in \mathbb{Z}$  mit  $0 \leq x \leq l$ . Sei  $y := g^x$
- private key:  $x$ , public key:  $y$
- $\text{encrypt}(m) = (u, v)$  mit  $u = g^r$ ,  $v = my^r$ ,  $r \in \mathbb{Z}$  zufällig
- $\text{decrypt}(u, v) = vu^{-x}$ , da:  $my^r(g^r)^{-x} = my^r(g^{-x})^r = my^r(y^{-1})^r = m$

## DLP auf elliptischen Kurven 1

- Auf Punktgruppen von (hyper-)elliptischen Kurven ist das DLP "besonders schwer"
- $E : y^2 = x^3 + ax + b$ , Punktmenge  $\{(x, y) \in K | y^2 = x^3 + ax + b\}$ ,  $K$  Körper
- Die Punktmenge, die die Gleichung erfüllt, ist eine Gruppe (Punktgruppe), und zwar mit einer speziellen Punktaddition und dem neutralem Element " $O$ " (s. Bild)
- Für kryptographische Zwecke werden Kurven über endlichen Körpern  $\mathbb{F}_q$  betrachtet

## Punktaddition auf elliptischen Kurven



## DLP auf elliptischen Kurven 2

Mathematische Fragestellungen z.B.

- Wieviel Punkte enthält  $E$  (über endlichen Körpern)?
- Für welche Gruppen und welche speziellen  $E$  ist das DLP nicht schwer bzw. besonders schwer?

## Literatur

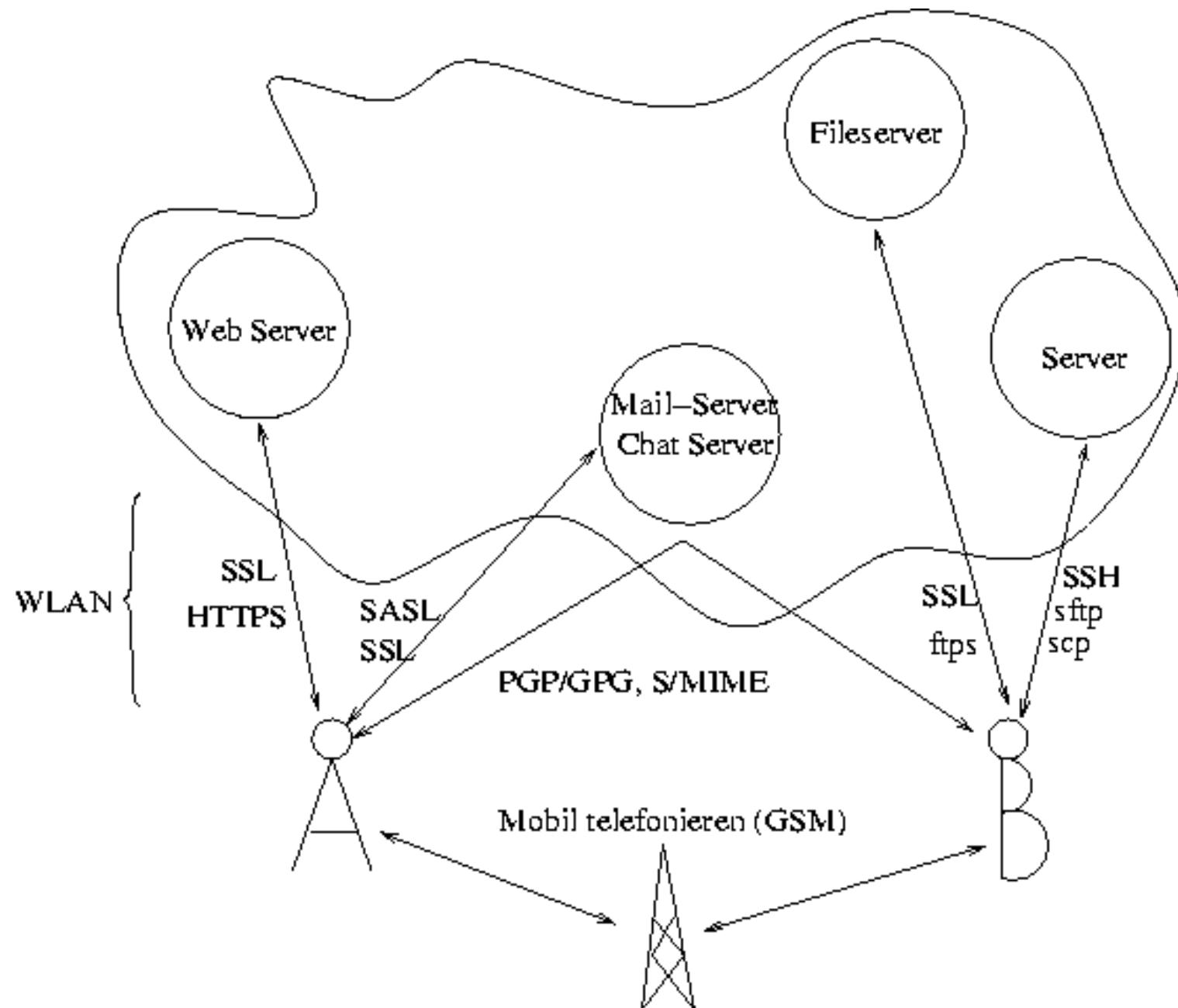
[http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt\\_Kryptologie](http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_Kryptologie)

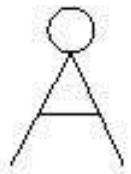
# Fragen

- Wie erzeuge ich mir ein Schlüsselpaar?
- Wie veröffentliche ich meinen öffentl. Schlüssel?
- Wie komme ich an Schlüssel meiner Kommunikationspartnerin?
- Woher weiss ich, dass der Schlüssel wirklich von ihr ist?
- Wie verschlüssle ich Daten und Datentransfer?
- Was mache ich, wenn ich meinen Schlüssel verloren habe oder er kompromittiert wurde?

Grundlagen:

Verschlüsseln  
Signieren  
Vertrauen  
Zertifizieren





Schlüsselpaar generieren



Schlüsselpaar generieren

öffentlicher Schlüssel per Mail



Schlüssel importieren  
Fingerprint generieren



per Telefon Fingerprint verifizieren

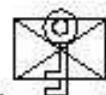


Öffentlichen Schlüssel  
auf Keyserver exportieren

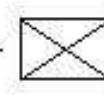
Keysigning Party

Öffentlichen Schlüssel  
signieren lassen

Schlüssel importieren



Mail verschlüsseln  
mit öff. Schlüssel von B



Mail entschlüsseln mit  
privatem Schlüssel

Mail entschlüsseln mit  
privatem Schlüssel



Antwort verschlüsseln mit  
öffentlichen Schlüssel von A

# Verschlüsseln, Signieren, Authentizität

- Verschlüsselung: Übertragung von Geheimnissen
- Signatur: Gewährleistung der Authentizität des Senders, der Integrität der Nachricht (wie Siegel)
- Wie stelle ich Authentizität des Gegenübers fest?
  - Direktes Vertrauen
  - Web Of Trust
  - Hierarchisches Vertrauensverhältnis

# Woher weiss ich, das Schlüssel zur Person gehört?

- Web of Trust
- PGP/GPG
- Key Signing Party
- ich vertraue Berta, weil ich Alice vertraue und Alice vertraut Berta
- Public Key Infrastructure mit hierarch. Zertifizierung
- SSL, S/MIME
- Behörden/Firmen
- zentrale Zertifizierungsinstanz

# Web Of Trust

- inwieweit traut man Eigentümer eines öff. Schlüssel korrekte Authentisierung zu  
(Unbekannt, kein/teilweises/volles Vertrauen)
- Vertrauen: Schlüssel ist von genügend gültigen Schlüsseln unterschrieben:
  - persönlich
  - von einem Schlüssel vollen Vertrauens
  - von drei Schlüsseln teilweisen Vertrauens
  - Pfad unterschriebener Schlüssel von Schlüssel K zurück zu eigenem Schlüssel führt < 5

# Zertifizierung

- Public Key Infrastructure: hierarchisches Netz von Zertifizierungsstellen
- Zertifizierungsstelle (Certification Authority, CA): Zertifikate ausstellen und veröffentlichen
- Cross-Signing zwischen CAs
- Public Keys einiger CAs im Browser
- Zertifikat enthält Infos über Schlüssel, Person, und digitale Unterschrift der zertifizierenden Behörde

# Gnu Privacy Guard (GPG)

Grundlagen  
Schlüssel erzeugen  
Schlüssel anzeigen  
Schlüssel exportieren  
Schlüssel importieren  
Schlüssel signieren  
Schlüssel widerrufen

# Grundlagen: PGP und GPG

- Was ist Pretty Good Privacy (PGP)?
  - Software für Verschlüsselung, Schlüsselerzeugung, Schlüsselverwaltung
  - standardisiert in OpenPGP
  - unabhängig von Datenformaten
  - kann symmetrisch und asymm. verschlüsseln
- Was ist Gnu Privacy Guard (GPG)?
  - basiert auf OpenPGP
  - vollständig kompatibel zu PGP, benutzt die gleiche Infrastruktur (Schlüsselserver usw.)

# So klappt's mit GPG

- Download: [www.gnupg.org](http://www.gnupg.org)
- bei Linux-Distributionen dabei
- gibt's auch für Windows
- Benutzung:
  - Kommandozeile
  - Frontends: GPA, KGpg / Kleopatra
  - aus der Applikation heraus (Mail, IM...)
- `man gpg`
- [www.gnupg.org/\(en\)/howtos/de/index.html](http://www.gnupg.org/(en)/howtos/de/index.html)

# Schlüssel erzeugen

```
jh@perdida:~> gpg --gen-key
gpg (GnuPG) 1.2.4; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.
```

```
Bitte wählen Sie, welche Art von Schlüssel Sie möchten:
(1) DSA und ElGamal (voreingestellt)
(2) DSA (nur signieren/beglaubigen)
(4) RSA (nur signieren/beglaubigen)

Ihre Auswahl? 1

Das DSA-Schlüsselpaar wird 1024 Bit haben.

Es wird ein neues ELG-E Schlüsselpaar erzeugt.

      kleinste Schlüssellänge ist 768 Bit
      standard Schlüssellänge ist 1024 Bit
      größte sinnvolle Schlüssellänge ist 2048 Bit

Welche Schlüssellänge wünschen Sie? (1024)

Die verlangte Schlüssellänge beträgt 1024 Bit

Bitte wählen Sie, wie lange der Schlüssel gültig bleiben soll.

      0 = Schlüssel verfällt nie
      <n> = Schlüssel verfällt nach n Tagen
      <n>w = Schlüssel verfällt nach n Wochen
      <n>m = Schlüssel verfällt nach n Monaten
      <n>y = Schlüssel verfällt nach n Jahren

Wie lange bleibt der Schlüssel gültig? (0)

Key verfällt nie.

Ist dies richtig? (j/n) j
```

# Schlüssel erzeugen (2)

Sie benötigen eine User-ID, um Ihren Schlüssel eindeutig zu machen; das Programm baut diese User-ID aus Ihrem echten Namen, einem Kommentar und Ihrer E-Mail-Adresse in dieser Form auf:

"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Ihr Name ("Vorname Nachname"): Jutta Horstmann

E-Mail-Adresse: jh@weltraumsofa.de

Kommentar: Workshop 3

Sie haben diese User-ID gewählt:

"Jutta Horstmann (Workshop 3) <jh@weltraumsofa.de>"

Ändern: (N)ame, (K)ommentar, (E)-Mail oder (F)ertig/(B)eenden? █

# Schlüssel erzeugen (3)

Sie benötigen eine Passphrase, um den geheimen Schlüssel zu schützen.

gpg: GPG-Agent ist in dieser Sitzung nicht vorhanden

Wir müssen eine ganze Menge Zufallswerte erzeugen. Sie können dies unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole irgendetwas tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.

Wir müssen eine ganze Menge Zufallswerte erzeugen. Sie können dies unterstützen, indem Sie z.B. in einem anderen Fenster/Konsole irgendetwas tippen, die Maus verwenden oder irgendwelche anderen Programme benutzen.

Öffentlichen und geheimen Schlüssel erzeugt und signiert.

Schlüssel ist als uneingeschränkt vertrauenswürdig gekennzeichnet.

pub 1024D/38AF4086 2004-12-22 Jutta Horstmann (Workshop 3) <jh@weltraumsofa.de>

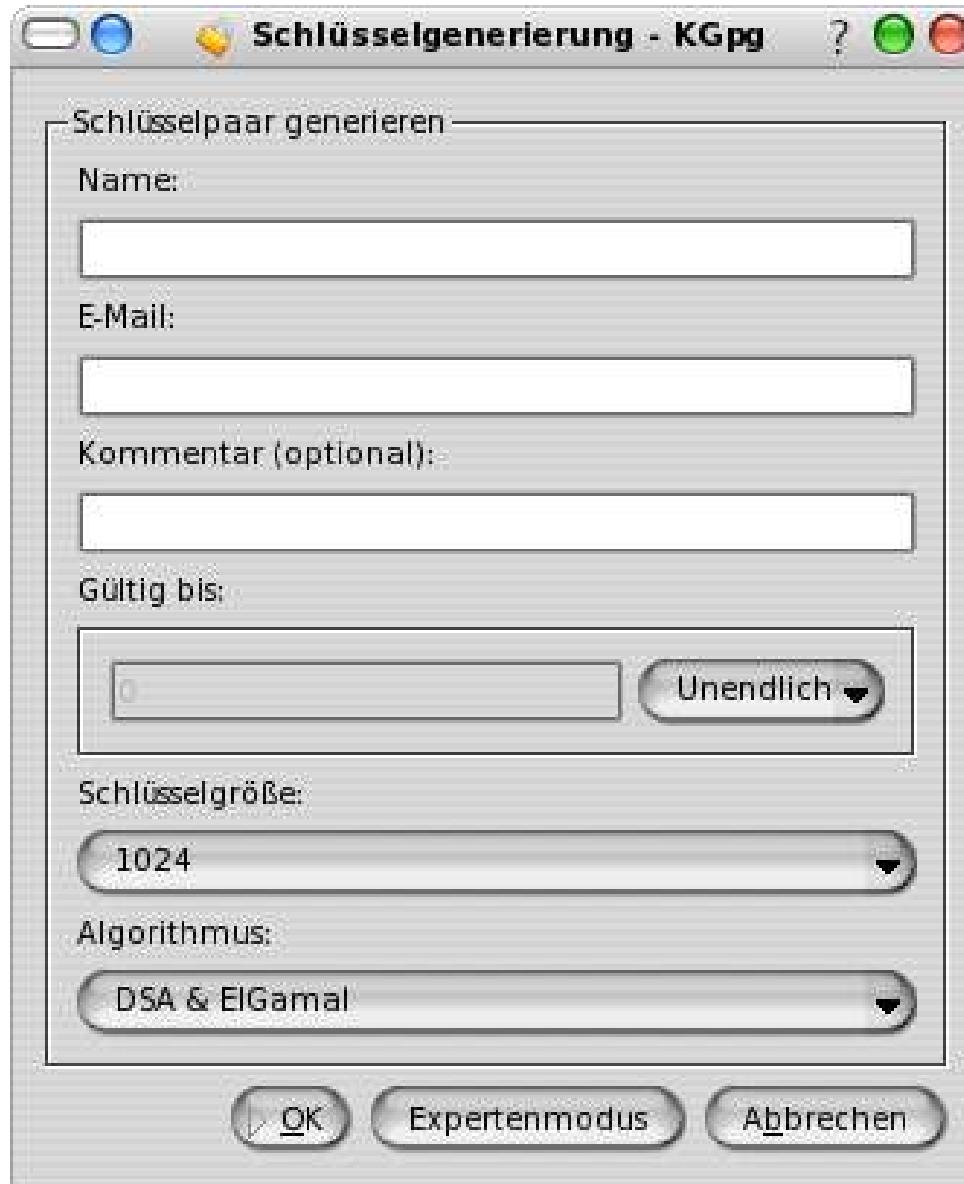
Schl.-Fingerabdruck = 1C1C DE52 7558 8337 AD58 AED9 F72E DB13 38AF 4086

sub 1024g/8CF15E14 2004-12-22

# Schlüssel erzeugen (4)

- Passphrase (Mantra):
  - symmetrische Verschlüsselung des privaten Schlüssels (einzige Sicherung!)
  - nicht zu kurz, Sonderzeichen...
- Zum Erzeugen von Schlüsseln muss “Zufall” generiert werden
  - Bsp. RSA: Man wählt **zufällig** zwei grosse Primzahlen p und q sowie ein **zufälliges** e

# Schlüssel erzeugen mit KGpg



# Schlüssel anzeigen

```
jh@perdida:~> gpg --list-keys
/home/jh/.gnupg/pubring.gpg
-----
pub 1024D/841A30D4 2001-06-05 Jan.Muehlig <Jan.Muehlig@relevantive.de>
uid                               Jan.Muehlig <ICQ:43905743>
sub 2048g/31727FD7 2001-06-05

pub 1024D/EF2EC201 2004-12-12 Jutta Horstmann <jh@weltraumsofa.de>
sub 1024g/D2CCDE85 2004-12-12

pub 1024D/1479D280 2004-12-17 Jutta Horstmann (Workshop 2) <jh@weltraumsofa.de>
sub 1024g/68716A29 2004-12-17
```

The screenshot shows the interface of the KGpg key manager. The window title is "Schlüsselverwaltung - KGpg". The menu bar includes "Datei", "Bearbeiten", "Ansicht", "Schlüssel", "Gruppen", "Einstellungen", and "Hilfe". Below the menu is a toolbar with icons for search, add, delete, and other key management functions. A search bar labeled "Suche:" is present. The main area displays a table of keys:

Name	E-Mail	Vertrauen	Gültigkeitsende	Größe	Erstellung	Kennung
+ Jan.Muehlig	Jan.Muehlig@relevantive.de	[empty]	Unbegrenzt	1024	05.06.2001	0x841A30D4
+ Jutta Horstmann	jh@weltraumsofa.de	[green]	Unbegrenzt	1024	12.12.2004	0xEF2EC201
+ Jutta Horstmann (Workshop 2)	jh@weltraumsofa.de	[green]	Unbegrenzt	1024	17.12.2004	0x1479D280

At the bottom left is a button for "Geheimes Schlüsselpaar" and at the bottom right is a status bar showing "3 Schlüssel, 0 Gruppen".

# Schlüssel exportieren

```
jh@perdida:~> gpg --export --armor Workshop1
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.4 (GNU/Linux)

mQGiBEG92bkRBADJbNzbvayA8IQNFM6Sy50Sd8ADY4hyjElXpIrzuViBB9dIbR3Z
MnsdYx6jp/EgqxNzsIwiNh0g1gNhdfhJ1IvSwhYjdiDTuIWgrw/aJhGw1oEIniOK
VQyNG4cCELXBfsGCrxgiYTX1hph5URGEqsOnKYIEBImj4MwKp669ph6IfwCgwR/R
GbT19n5RW3wcLQqY1S20hCsEA14zKLGSXpNv6bPB3YmlYQubsrbQSAbh14aYMX0m
1WJE5xQq6ywU2KfMZ43svTdRIz1hZi5dvhR4N8COnE+WlPhIiiYgFZzKtpVl1YP
8GJU+y3Y7/evFUUe0ampm9k4ijhoJBPPn1ak0sQiRh6RpQ26fmxMGeaVjsf6/RPe8
cXzsBACqh+AVM1F8VyWi9ieSOHkVh+kL9kPhkaBfh5kuXfKK8cj0nuPSQkwgalt2
GvpGN1K+FGva6ZHoeWEHzT6J1z9kWxwEj/eXUw1nv5aBb8ATelaCu5HWcFKQkPcFc
GH686nU6KDijrY+09w0N5dnIWOh5UicyHSeWE9S/5oKrlxBVR7QwSnV0dGEgSG9y
c3RtYW5uIChXb3Jrc2hvcDEpIDxqaEB3ZWx0cmF1bXNvZmEuZGU+iF4EExE CAB4F
AkG92bkCGwMGCwkIBwMCAxUCAwMWAgECHgECF4AACgkQw9dKoo/cHeyeYwCgijr7
WEWjYPXjUQ/bloCV7+/vtf0Anjwhta6AJ/urFx/zsSzgiTtjeo2VuQENBEG92bsQ
BADyj721KsEzE0/YMBUpj9zlzS5G5jEpRYalwMRggpy+sJw/RIRack8+BOH478MH
qtZ1Edsd5S4IzhccoYmFKb4c5ztULL1cDaPIp18zFweZE4MnUrUhTkmLp9VP7stb
DwBf7mzrCf+0Sw8dPUT42JNyhdTvZ05VwKBLQSB8UF/t2wADBgP/VyWhG3RBwu8p
XTL7fwbsnFAw/9vvKyBD0BvhidaO+sZeMzr1CTqUoUF6E8FAxit9JF6kMJWHQ1oU
QfjiBb2akWLwfz8SrxMEUKnfKplIkI9LDUEQPIA5e4bKjh04ZAwpulMhe6ITxTgC
Oip+P03j/lxtd4+LzuAH88ssvAkFEC+ISQQYEQIACQUCQb3ZuwIbDAAKCRDD10qi
j9wd7Bd1AJ9a/04kd2Qdwd0WMLncDDr6NLANOgCePUOHB5F+yX6Q7SzSTBXfNA0d
t3Q=
=yANE
-----END PGP PUBLIC KEY BLOCK-----
jh@perdida:~> █
```

# KGpg: Schlüssel exportieren



# Schlüssel bekannt machen

- Keyserver
  - gpg --send-keys [names] --keyserver [name]
  - z.B. hkp://wwwkeys.eu.pgp.net
  - eigenen Schlüssel vom Keyserver aktualisiert holen (unterschrieben)
  - unterschriebene Schlüssel hochladen
- versenden per Mail
- eigene Website
- Visitenkarte (Fingerprint)

# GPG – importieren

- Importieren des Schlüssels einer Kommunikationspartnerin:
  - gpg --import [Datei]
  - ohne Dateiname: von Kommandozeile
  - z.B. aus Mail: von Enigmail unterstützt
- Schlüssel anhand des Fingerprints verifizieren

```
jh@perdida:~> gpg --fingerprint EF2EC201
pub 1024D/EF2EC201 2004-12-12 Jutta Horstmann <jh@weltraumsofa.de>
Schl.-Fingerabdruck = E0EE FB6B 3442 5A60 FE09 80CF 8E8C 24FA EF2E C201
```

# GPG: Schlüssel signieren (1)

- Fremdschlüssel:
  - ist der Schlüssel überprüft, kann man ihn signieren
  - mit “check” gucken, wer den Schlüssel sonst schon beglaubigt hat
- eigener (öffentlicher) Schlüssel:
  - könnte manipuliert werden
  - daher: signieren (Eigenbeglaubigung) mit privatem Schlüssel (automatisch)

# gpg --edit-key UID

Befehl> sign

Wirklich alle User-IDs beglaubigen? j

pub 1024D/841A30D4 erstellt: 2001-06-05 verfällt: niemals Vertrauen: f/-  
Haupt-Fingerabdruck = 161D F793 E2D5 B6D2 F9A5 2709 4065 A1E8 841A 30D4

Jan.Muehlig <Jan.Muehlig@relevantive.de>

Jan.Muehlig <ICQ:43905743>

Wie genau haben Sie überprüft, ob der Schlüssel, den Sie jetzt beglaubigen  
wollen, wirklich der o.g. Person gehört?

Wenn Sie darauf keine Antwort wissen, geben Sie "0" ein.

- (0) Ich antworte nicht.Daten entschlüsseln (Voreinstellung)
- (1) Ich habe es überhaupt nicht überprüft.
- (2) Ich habe es flüchtig überprüft.
- (3) Ich habe es sehr sorgfältig überprüft.

Ihre Auswahl? ('?' für weitere Informationen):

# KGpg: signierter Schlüssel

Schlüsselverwaltung - KGpg

Datei Bearbeiten Ansicht Schlüssel Gruppen Einstellungen Hilfe

Suche:

Name	E-Mail	Vertrauen	Gültigkeitsende	Größe	Erstellung	Kennung
Jan.Muehlig	Jan.Muehlig@relevantive.de	[white]	Unbegrenzt	1024	05.06.2001	0x841A30D4
+ ElGamal Unterschlüssel		[green]	Unbegrenzt	2048	05.06.2001	0x31727FD7
+ Jan.Muehlig	ICQ x3a43905743	[green]	-	-	-	-
+ Jan.Muehlig	Jan.Muehlig@relevantive.de	-	Unbegrenzt	-	05.06.2001	0x841A30D4
+ Jutta Horstmann	jh@weltraumsofa.de	-	Unbegrenzt	-	27.12.2004	0xEF2EC201
+ Jutta Horstmann	jh@weltraumsofa.de	[green]	Unbegrenzt	1024	12.12.2004	0xEF2EC201
+ Jutta Horstmann (Workshop 2)	jh@weltraumsofa.de	[green]	Unbegrenzt	1024	17.12.2004	0x1479D280

3 Schlüssel, 0 Gruppen

# Key Signing Party

- HOWTO:  
<http://www.cryptnet.net/fdp/crypto/gpg-party/gpg-party.de.html>
- Ziel: gegenseitig Schlüssel signieren
- Web Of Trust vertiefen
- Identität der Person verifizieren (Ausweis)
- Schlüssel verifizieren und signieren
- signierten Key an Schlüsselbesitzer oder Keyserver exportieren

# Key Revocation (1)

- gpg --gen-revoke
- gpg --output revoke.asc --gen-revoke
- Widerrufsurkunde ausdrucken und an sicherem Ort aufbewahren
- jeder kann damit den Schlüssel ungültig machen

# Key Revocation (2)

```
jh@perdida:~> gpg --gen-revoke EF2EC201

sec 1024D/EF2EC201 2004-12-12 Jutta Horstmann <jh@weltraumsofa.de>

Ein Widerrufszertifikat für diesen Schlüssel erzeugen? (j/N) j
Grund für den Widerruf:
 0 = Kein Grund angegeben
 1 = Hinweis: Dieser Schlüssel ist nicht mehr sicher
 2 = Schlüssel ist überholt
 3 = Schlüssel wird nicht mehr benutzt
 Q = Abbruch
(Wahrscheinlich möchten Sie hier 1 auswählen)
Ihre Auswahl? 0
Geben Sie eine optionale Beschreibung ein. Beenden mit einer leeren Zeile:
> Key no longer in use
>
Grund für Widerruf: Kein Grund angegeben
Key no longer in use
Ist das richtig? j
```

# Key Revocation (3)

Sie benötigen eine Passphrase, um den geheimen Schlüssel zu entsperren.

Benutzer: "Jutta Horstmann <jh@weltraumsofa.de>"

1024-Bit DSA Schlüssel, ID EF2EC201, erzeugt 2004-12-12

gpg: GPG-Agent ist in dieser Sitzung nicht vorhanden

Ausgabe mit ASCII Hülle erzwungen

Widerrufszertifikat wurde erzeugt.

Bitte speichern Sie es auf einem Medium welches sie wegschliessen können; falls Mallory (ein Angreifer) Zugang zu diesem Zertifikat erhält, kann erIhren Schlüssel unbrauchbar machen. Es wäre klug, dieses Widerrufszertifikat auch auszudrucken und sicher aufzubewahren, falls das ursprüngliche Medium nicht mehr lesbar ist. Aber Obacht: Das Drucksystem kann unter Umständen eine Kopie anderen Nutzern zugänglich machen.

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: GnuPG v1.2.5 (GNU/Linux)

Comment: A revocation certificate should follow

# Praxis

Dateien verschlüsseln  
Mailverkehr verschlüsseln

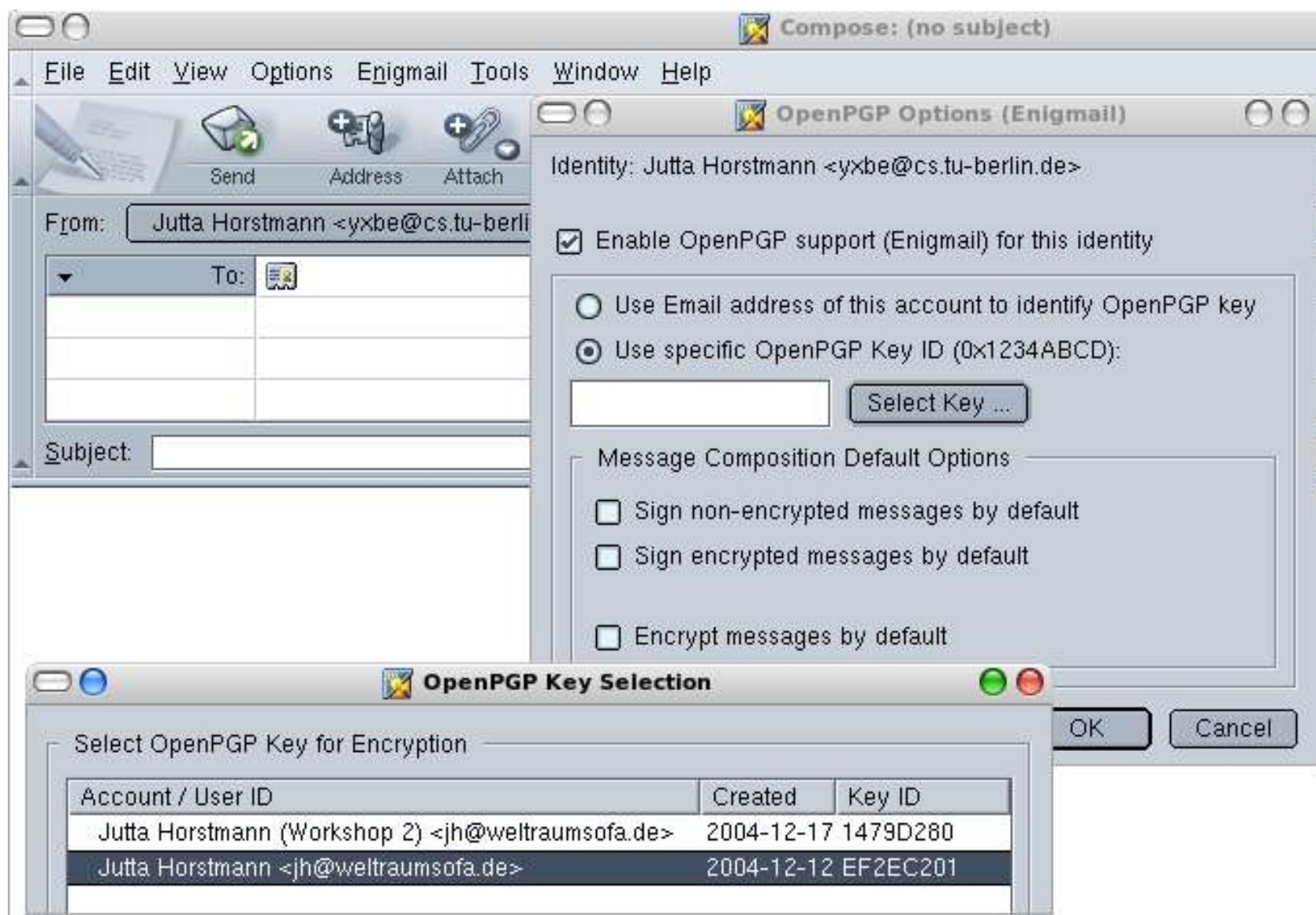
# Verschlüsseln von Dateien

- Asymmetrisch verschlüsseln:
  - gpg --output test.gpg --encrypt --recipient EF2EC201 test.txt
  - bei recipient auch mehrere Empfänger
- Asymmetrisch entschlüsseln:
  - gpg --output test.klar --decrypt test.gpg
  - Passphrase eingeben
- Symmetrisch verschlüsseln:
  - gpg --output test.sym --symmetric test.txt
  - Passphrase zum Ver- und Entschlüsseln

# Mail verschlüsseln

- Mailprogramme:
  - Linux: Enigmail (Mozilla/Thunderbird), Evolution, KMail, mutt, Sylpheed...
  - Win: Enigmail (Mozilla/Thunderbird), Outlook Plugin, EudoraGPG...
  - Mac: Enigmail (Mozilla/Thunderbird), Apple Mail Plugin...
- Beispiel Enigmail:
  - <http://enigmail.mozdev.org/>

# Enigmail einrichten



# Enigmail: Verschlüsselt senden



# Enigmail: Verschlüsselt empfangen

▽ **Subject:** 21C3 Keysigning (key: 0xEF2EC201)  
**From:** gpgbot-21C3@kuehne.cn  
**Date:** 22.12.2004 15:09  
**To:** jh@weltraumsofa.de

-----BEGIN PGP MESSAGE-----

```
hQE0A+9KvP/SzN6FEAP/Z8PLVIqwEiT
0POH2L9cCEYuYZEcRNM90jnhrHN1luA
tWx2AsYH8fFf/2n9HTuBlhZzGa7f4gZ
/2ciWo7cQIvows5Jvb51UyBuu8WUEmC
oVjJeKxnyFSUZYWZiYlpVrg7qYpfSbQ
HGvc2yak/h6TV+tZMUWYjAUZfnlewch
m5xwVN+DH/Xfkuy/IJ9/z1WLVYED3BW
z3rIfjIXDhDaneNNuMWIQUVh6nNbLCsZa35FtxqWM5zP/bwBmfYp1tdkT8LcmBo0
EFFo8YHreKBs0GjiyTI14ZC79ttAqEyNiLdD/j41xMUDchaxXr3ezjqIfJ14NHqx
LF30tN29B389xzHjVlFoiyYwY5kM4LUFCKSmu9f/chQUJcaMjP+oo3fZMsdwHxkj
pY1Kmeg/3EZ3ddYY0PiKOX0mlX5pR2nht1wBbqlLUV4q3aZnHypxfosalsNRGseKb
bxzDg7eJ1ckzuBqBTA77zgf/RqjgMn2pnwXSylB38C0ozYzUF0DPZrFjMqnZJvNI
ailBGV/xlsEI09y8wRV9c1U0F1R8cZbfnr8AF0BpaWQquVoeRPABiob8bZQ10BhT
oHfMhdUgFKlzpZtbgKnJTeBYUtRf7yDECes2CHgx99J3yqCd45zXhFA6w0gfx5pr
COOrFiuZt5wLejhfkYIXpBsomMp1Vm64Bgy/xS4MeZNyJb/xXZiKdXn0v9a2zJ786
0qrYXTzHFo/CQT6Zmk019yBL6d1U5J0g2dChfNGL18pbfMzjug5+KyJNqXhD03T9
```



The screenshot shows a Mac OS X style dialog box titled "Enigmail". It contains a question mark icon, a text input field, and a checkbox labeled "Remember for 5 idle minutes". There are "OK" and "Cancel" buttons at the bottom.

# Sonstige Datenströme

Grundlagen SSL  
Datenverkehr im WWW

Mailserver: Sichere Authentifizierung

SSH  
FTP

# SSL

- Zunächst zur Absicherung von HTTP (“https”)
- Später generalisiert zu TLS
- Sicherheitsprotokoll der Transportschicht
- Sichert beliebige Anwendungen, die auf TCP aufsetzen
- Meist Server-Authentisierung
- X.509 Zertifikate (wichtigster Standard)
- verschiedene kryptographische Verfahren mit variablen Schlüssellängen
- Open Source: OpenSSL, GnuTLS

# Datenübertragung im WWW

- Ist der Server der, für den ich ihn halte? (z.B. wirklich [www.berliner-sparkasse.de](http://www.berliner-sparkasse.de)?)
  - Server hat zertifizierte Schlüssel
- Aufbau einer SSL-Verbindung:
  - Server sendet zertifizierten Public Key
  - Browser überprüft Key und Zertifikat
  - nicht ok: Warnmeldung von Browser
  - okay - dann verschlüsselter Austausch zwischen Browser und Server
- keine Client-Authentisierung

# Mail: SSL, STARTTLS

- Serverseitiges Angebot für POP, IMAP, SMTP
- Im Mail-Client anschalten!
- StartTLS (Start Transport Layer Security): TLS für SMTP
- es werden auch Inhalt und Attachment (der gesamte Datenstrom) der Mail verschlüsselt! (aber nur von Punkt zu Punkt)
- SMTP AUTH: nur Login/PW verschlüsselt

# Secure Shell (SSH)

- Eigenes Protokoll unabhängig von SSL
- Kommandozeilenzugriff auf entfernten Rechner
- sichere authentifizierte und verschlüsselte Verbindung zwischen zwei Rechnern über ein unsicheres Netzwerk
- X11-Sitzungen und andere TCP/IP-Verbindungen über ssh tunneln
- Sendet RSA Key Fingerprint, der geprüft werden und bei späteren Logins verglichen werden kann

# File Transfer (FTP)

- SCP, SFTP: nutzen SSH zum Datentransfer
- FTP mit SSL (FTPS)
  - Serverseitig: SSL nutzen (Zertifikat generieren, TLS im Server anschalten, z.B. proftp, vsftpd)
- Clients/Frontends:
  - Linux: Gftp (scp, sftp, ssl), Konqueror (sftp, ssl)
  - Win: WS FTP (SSL), Putty (SFTP), WinSCP
  - Mac: Fugu (SFTP, SCP)

# Verschlüsselung in Funknetzen

Wireless LAN  
Mobiles Telefonieren

# WLAN

- Wireless Equivalent Privacy (WEP):
  - meiste Produkte: nur 40bit Schlüssel
  - gezielte Modifikationen an Nachrichten, Mitlesen, Schlüsselberechnung sind möglich
  - kein Schlüsselmanagement, es gibt einen Shared Key für das gesamte Netzwerk
  - nur Client authentisiert, nicht AP oder Nutzer
- Lösungen:
  - stattdessen SSL, SSH, IPSec o.ä. verwenden
  - VPN hinter Access Point aufbauen

# WLAN (2)

- 2002: WPA (Wi-Fi Protected Access)
  - Verschlüsselung: TKIP (Temporal Key Integrity Protocol), basiert auf WEP, behebt dessen größte Fehler
  - Geräte per Firmware-Update aufrüsten
- 2004: Standard IEEE 802.11i:
  - TKIP
  - WRAP (Wireless Robust Authenticated Protocol) basierend auf AES (Advanced Encryption Standard) und CCMP (vollständig neues Protokoll ohne Verwandtschaft zu WEP)
  - neue AP-Geräte notwendig

# Mobil Telefonieren

- Drahtlose Übertragung leichter anzuzapfen
- Gerät muss immer neu authentifiziert werden, wenn es den Netzzugangspunkt wechselt
- Aufenthaltsort des Gerät u.U. ähnlich interessant wie Inhalt des Gesprächs
- Location Privacy ist nicht gegeben

# Mobil Telefonieren (2)

- GSM:
  - an sich immer Verschlüsselung - zwischen Handy und Basisstation
  - Provider kann Verschl. jederzeit deaktivieren
- Es gibt Software zur End-zu-End-Verschlüsselung
- anonyme Pre-Paid-Karten nutzen
- Verschlüsselungssoftware für die **Daten** auf dem Handy (Fotos, Notizen...)

# Noch Fragen?

